Since SP Devices’ high-speed digitizers produce a higher data rate than most host computers can handle, on-board feature extraction is often necessary to reduce the data rate. The methods of data reduction and the methods of transferring data to the host PC are closely connected and can be combined in many ways to meet various systems requirements. This application note is a guide to how to set up the data transfer and combine it with on-board feature extraction (data reduction) to meet the requirements of the application.
Table of Contents

1 Introduction ............................................................................................................. 4
2 Data reduction; motivation ...................................................................................... 6
  2.1 Introduction to SP Devices’ high speed digitizers .............................................. 6
  2.2 Data rate from the ADCs to the FPGA on the digitizer .................................. 6
  2.3 Connecting to a host PC ................................................................................... 7
3 Data reduction methods .......................................................................................... 7
  3.1 Introduction ....................................................................................................... 7
  3.2 Type 1: Triggered record acquisition for pulse data .......................................... 7
  3.3 Selecting multi-record or triggered streaming ................................................... 9
  3.4 Type 2: Bandwidth reduction for RF/IF data .................................................... 12
4 Multi-record mode .................................................................................................. 13
  4.1 ADQ DRAM as data buffer .............................................................................. 13
  4.2 Sequence of operation ..................................................................................... 13
  4.3 Transferring the records to the host PC ........................................................... 15
  4.4 Memory dump and off line processing .............................................................. 15
  4.5 Example of ADQ412-4G multi-record ............................................................... 17
5 Triggered streaming ............................................................................................... 17
  5.1 FPGA block RAM as buffer .......................................................................... 17
  5.2 Transfer data from ADQ Block RAM to host PC ............................................. 19
  5.3 Example of ADQ108 using triggered streaming .............................................. 20
6 Data reduction for continuously streaming systems ............................................. 20
  6.1 System ............................................................................................................ 20
  6.2 Decimation ..................................................................................................... 20
  6.3 Buffer .......................................................................................................... 20
  6.4 Example of continuously streaming system RF system ................................... 20
7 Data link from digitizer to host PC ...................................................................... 21
  7.1 Data packet sizes .......................................................................................... 21
  7.2 Command sequence for data transfer ............................................................... 24
8 Setting up the host PC for streaming ................................................................. 27
  8.1 Introduction ................................................................................................... 27
  8.2 Principle of operation .................................................................................... 27
  8.3 Setting buffer size ......................................................................................... 27
  8.4 Re-allocate RAM .......................................................................................... 27
  8.5 Example on allocating RAM ........................................................................ 27
9 Pre-allocation application ...................................................................................... 29
  9.1 Introduction ................................................................................................... 29
  9.2 Starting the pre-allocation application ............................................................. 29
  9.3 Setting RAM size .......................................................................................... 29
  9.4 Status check .................................................................................................. 29
10 Consuming the data in the host PC .................................................................... 29
  10.1 Extracting parameters .................................................................................. 29
  10.2 Storing raw data to disk ................................................................................ 29
Appendix A Reference guide to ADQAPI_preallocate.exe ...................................... 30
Appendix B Parameters for the different digitizer models ...................................... 32
Appendix C ADQ DRAM format .............................................................................. 33
  C.1 Data records in the ADQ DRAM ................................................................. 33
  C.2 Packing data in the ADQ DRAM ................................................................. 33
  C.3 Unpacking data from the ADQ DRAM ....................................................... 34
Appendix D USB freezing ....................................................................................... 34
Appendix E Acquisition modes example code ....................................................... 35
General note about data transfer speed
The ADQ V6 digitizer family supports up to 3.6 GB/s sustained data transfer rate to the host system RAM. To achieve the best performance, one must evaluate high performance requirements on multiple components in the host system.

To achieve a high PCI express throughput, the system must support a PCI express payload size of 256 bytes. Whether a system supports this depends on multiple components, including CPU, chipset and BIOS. If there are PCI express switches in the PCI express hierarchy between the CPU and the ADQ digitizer, these switches also have to support the payload size of 256 bytes.

The data transfers also place high load on RAM performance, especially if data will be copied or manipulated while transferring data continuously. The system must handle the RAM load from both running applications, operating system, receiving ADQ data and potentially additional read/writes from any processing or further buffering.

Finally, the CPU must handle the load from transfers. Transferring large blocks of data from ADQ to host system RAM is a moderate load on most CPUs, since data is copied by a DMA engine, and the CPU must only manage these transfers. However processing of data, or additional data accesses are very intensive for most CPUs. Most cannot process data real-time without utilizing optimized multi-threaded routines.

General note about units
Note the different definitions of the prefix k (kilo) versus Ki (kibo), M (mega) versus Mi (mibi), and G (Giga) and Gi (gibi).

- All data rates are related to sample rate and given in the base $10^X$ ($10^3$, $10^6$, and $10^9$) and thus have the metric k, M and G.

- All data sizes are related to addressing with binary numbers and given in the base $2^X$ ($1024^1$, $1024^2$, and $1024^3$) and thus have the binary prefix Ki, Mi, Gi.

(In the unlikely event of a typo in this document, where the wrong prefix seems to be used, always rely on the type of data.)
1 Introduction

The data transfer from the digitizer to the host PC can be done in several ways depending on the typical characteristics of the user’s system. This application note describes various solution for different use cases. The data transfer properties is heavily dependent of the methods of data reduction inside the digitizer. The data transfer is thus described from a data reduction perspective.

The description in this application note is based on the hardware view in Figure 1. The logical view in of the system is in Figure 2 and in Table 1. It has six main blocks, which put different restrictions on the solution and thereby influence the choice of data transfer method. There are four blocks where the data rate influences the system; ADC, data reduction, link to host PC and data consumption in the processing unit. In between are two blocks that determines the sizes of the data records; data buffer in the digitizer and RAM in the host PC. Table 1 contains links to the corresponding section in this application note.

![Figure 1: Physical view of the system model in this application note.](image1)

![Figure 2: Logical view of the system model in this application note. See Table 1 for description.](image2)
ADC samples data from a sensor and converts to a digital stream of data characterized by a sample frequency and a word length (number of bits per sample).

Rate The sensor is not included in the description. It is assumed that the selected digitizer suits the use case well.

Data reduction
Reduces the data rate of the real-time data inside the digitizer to match the rate on the link to the host PC and/or the data processing capacity of the host PC.

Rate This is done as either triggered acquisition, bandwidth reduction through filtering, or custom processing in the FPGA.

Data FIFO
This is a data buffer in form of a FIFO on the digitizer that handle the real-time data from the ADC and reshape it to packet data on the data link.

Size
This is Block RAM in the FPGA for streaming mode.

Data link
This is the physical connection to the host PC.

Rate PCIe or USB2.0

PC RAM
A data buffer in the host PC that receives packets of data from the data link.

Size
This is limited by the amount of PC RAM that is available for the application.

Table 1: System in this application note, see Figure 2.

1. When using DRAM, there may also be a speed limitation for some combinations of settings.
2. Writing data to the DRAM in combination with other activities in the PC may put a speed requirement on the DRAM access.
2 Data reduction; motivation

2.1 Introduction to SP Devices’ high speed digitizers

SP Devices’ digitizers are characterized by high sampling rate in combination with high resolution. The result is that the intrinsic data stream from the ADCs is, in most cases, higher than the data link to the host PC can handle. Thus, data reduction is required between the ADC and the link to the host PC. This section motivates why data reduction in the digitizer is necessary.

From a data transfer perspective, SP Devices’ digitizers can be grouped into ADQ V6 series and ADQ V5 series, see Table 2. The relevant parameters are common for the digitizers within the same group.

2.2 Data rate from the ADCs to the FPGA on the digitizer

Table 2 shows the total data rate from the ADCs into the FPGA on the digitizers. Note that this is an internal data connection between the ADC and the FPGA and not the data rate out from the digitizer.

<table>
<thead>
<tr>
<th>MODEL</th>
<th># CH</th>
<th>SAMPL. RATE [MSPS]</th>
<th>VERT. RES. [BITS]</th>
<th>TOTAL DATA RATE FROM ALL ADCS(^1) [GBITS/S]</th>
<th>BYTE-WISE DATA RATE (^2) [GBYTES/S]</th>
<th>SUSTAINED RATE PCIE (^3) 4 5 [GBYTES/S]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADQ14 digitizer family</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADQ14-2A</td>
<td>2</td>
<td>500</td>
<td>14</td>
<td>14</td>
<td>1.8</td>
<td>2</td>
</tr>
<tr>
<td>ADQ14-4C</td>
<td>4</td>
<td>1000</td>
<td>14</td>
<td>56</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>ADQ14-2X</td>
<td>2</td>
<td>2000</td>
<td>14</td>
<td>56</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>ADQ108</td>
<td>1</td>
<td>7000</td>
<td>8</td>
<td>56</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>ADQ208</td>
<td>2</td>
<td>4000</td>
<td>8</td>
<td>64</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>ADQ412</td>
<td>4</td>
<td>1000</td>
<td>12</td>
<td>48</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>ADQ412-3G</td>
<td>4</td>
<td>1800</td>
<td>12</td>
<td>86.4</td>
<td>10.8</td>
<td>14.4</td>
</tr>
<tr>
<td>ADQ412-4G</td>
<td>4</td>
<td>2000</td>
<td>12</td>
<td>96</td>
<td>12</td>
<td>16</td>
</tr>
<tr>
<td>SDR14(^6)</td>
<td>2</td>
<td>800</td>
<td>14</td>
<td>22.4</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>ADQ1600</td>
<td>1</td>
<td>1600</td>
<td>14</td>
<td>22.4</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>ADQ V6 digitizer family</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADQ112</td>
<td>1</td>
<td>1100</td>
<td>12</td>
<td>13.2</td>
<td>1.65</td>
<td>2.2</td>
</tr>
<tr>
<td>ADQ114</td>
<td>1</td>
<td>800</td>
<td>14</td>
<td>11.2</td>
<td>1.4</td>
<td>1.6</td>
</tr>
<tr>
<td>ADQ212</td>
<td>2</td>
<td>550</td>
<td>12</td>
<td>13.2</td>
<td>1.65</td>
<td>2.2</td>
</tr>
<tr>
<td>ADQ214</td>
<td>2</td>
<td>400</td>
<td>14</td>
<td>11.2</td>
<td>1.4</td>
<td>1.6</td>
</tr>
<tr>
<td>ADQ V5 digitizer family</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Data rates in ADQ V6 and ADQ V5 digitizers.

1. For multi-channel models, the total data rate equals the sum of the data rate of all channels.
2. Byte-wise data rate is relevant since the data words in a user application has to be 8 or 16 bits. This is thus the amount of data that is delivered to the user.
3. These figures put requirements on the PC hardware and also on the task scheduling in the PC. The values are to be confirmed for a specific configuration.
4. Sustained data rate means average data rate for an amount of data that is much larger than the size of the buffer of the digitizer.
5. ADQ V6 requires PCIe Gen2 x8. ADQ V5 requires PCIe Gen 1 x4.
Data Transfer from ADQ to host PC Application Note

Preliminary

2.3 Connecting to a host PC

The ADQ V5 digitizer family support PCIe Gen1 by 4 lanes with a data rate of 10 Gbit/s on the physical link, including packet header and coding. In practice the link reaches 790 MBytes/s sustained data rate.

The ADQ V6 digitizer family supports PCIe Gen1 by 4 and also PCIe Gen2 by 8 lanes with a data rate of 40 Gbit/s. In practice the digitizer reaches 3.2 GBytes/s sustained data rate. The data in Table 2 show that the ADCs can produce a much higher data rate than the link to the host PC can transfer. It is thus concluded that some type of data reduction inside the digitizer is required.

The digitizers also support a USB 2.0 high-speed data link. Most of text in this document is also applicable to the USB interface. However, the data rate is then effectively limited to 25 MBytes/s sustained for all units.

3 Data reduction methods

3.1 Introduction

Data reduction is categorized into two different types; triggered record acquisition, Section 3.2, and bandwidth reduction, Section 3.4.

3.2 Type 1: Triggered record acquisition for pulse data

Triggered record acquisition is typically used in systems with pulsed data. At each trigger event, a record of data is acquired at full speed, Figure 3. The data reduction is achieved by discarding data between the records. The average data rate is thus reduced.

To use this data rate reduction method, there has to be a data buffer in the digitizer, that is, a FIFO. The full speed data is temporarily stored in the FIFO on the digitizer and transferred to the PC on a lower average speed. The requirement on the transfer capacity is then calculated as:

\[
\text{Average data rate} = \text{Trigger frequency} \times \text{Record length}
\]

\[\text{(1)}\]

Figure 3: Trigger and record in time domain.

1. The digitizers are available with USB 2.0 and PCIe interface. However, the values in the examples assume a PCIe connection. The PXIe and Micro-TCA form factors use PCIe for data transfer.

6. Valid for analog inputs only. The Arbitrary Waveform Generator part of SDR14 is excluded from this discussion.
For some applications, there are bursts of very fast triggers. Then a burst of records are stored in the data buffer before it transferred to the host PC, see Figure 4. This view is also valid if the trigger repetition rate (frequency) is not constant, which is the case for a data driven trigger such as level trigger. For this situation, the average data rate as well as the peak buffering capacity are critical parameters.

![Figure 4: Burst trigger and record in time domain.](image)

The standard firmware in the digitizer supports triggering as a data reduction method. It is also possible to combine that with a custom data reduction in the FPGA. The custom implementation may be energy calculations, peak detection, FFT, etc. Then, the standard ADC data in the record may be changed to any type information. It is up to the user’s software to interpret the information. Figure 5 illustrates how a custom data record can be implemented in the FPGA firmware.

![Figure 5: Example of custom data reduction in a pulse data system.](image)

One key component in the signal chain is the data buffer inside the digitizer. To allow for a large variety of system parameters, there are two different buffers with different characteristics; FPGA Block RAM and ADQ DRAM. See Figure 6 and Table 3 for a description of the two buffers.

---

1. The FPGA is accessed via the ADQ Development Kit, which is a firmware development kit that is purchased separately. The ADQ Development Kit contains the necessary standard digitizer functions and a user area for custom code.
2. Note that the buffer on the digitizer is not the same as a data record. The buffer is the physical memory on the digitizer that sets the maximum allowed delay from the sample time until the time when data is transferred to the host. A record is a set of consecutive samples acquired at one trigger event.
3. The ADQ DRAM is a physical DRAM on the ADQ digitizer. It is connected to the FPGA and can accept real-time data from the ADCs. The term ADQ DRAM is used to distinguish from the DRAM in the PC.
The two different data buffers also corresponds to two key operational modes of the digitizer; multi-record and triggered streaming. These terms will be used throughout the document. The type of buffer is selected depending on the mode of operation:

### Multi record mode
A multi-record mode is a scheduled system, where a set of records are first acquired, then transferred to the host PC. In this mode, the on board ADQ DRAM is used for larger records or a large set of records. The advanced flow control of the ADQ DRAM, however, limits the flexibility. Therefore the multi-record mode is scheduled as first acquire a set of records and then transfer all of them to the host PC. See Section 4 for further description.

### Triggered streaming mode
The triggered streaming mode means that the data follows a type of virtual circuit switching connection; acquire – transmit to host PC. Data streams through the system. The block RAM in the FPGA is used as FIFO this mode. The limiting factor is the size of the block RAM, which limits the maximum length of the data record. See Section 5 for further description.

### 3.3 Selecting multi-record or triggered streaming
The two modes, multi-record and triggered streaming are complementing each other. This section is guiding how to select operational mode. For deeper understanding, read Section 4 and Section 5.

The key differentiating parameter is the size of the FIFO. The multi-record mode will have a large FIFO, whereas the triggered streaming mode will have a FIFO 1000 times smaller. The multi-record mode can thus accept larger peak data rates than the triggered streaming. This can, for example be used in a burst recording system.
The advantage of triggered streaming is the system’s timing. The multi-record rely on polling flags, and controlling the flow from the software, Figure 7. The triggered streaming can be set-up to be completely driven by the triggers, Figure 8.

Taking the methods from Figure 7 and Figure 8 and combining it with the typical key parameters of the respectively methods give the use-cases in Figure 9 and Figure 10. Selecting the proper method is mapping the application to one of these structures.
Multi record mode

**Figure 9** (a) is common parameters in this example. All parameters in **Figure 9** have typical best case values. See **Table 6** for parameters for the different models.

In **Figure 9** (b) is the typical multi-record use-case. A set of records are acquired and then sent to the host PC. The total amount of data is large and thus stored in DRAM.

**Figure 9** (c) is a data driven application where analog activity activates the recording, that is, level trigger. The peak load is large, so the DRAM has to be used.

**Figure 9** (d) illustrates long records and long pre-triggers.

---

### Table 6

<table>
<thead>
<tr>
<th>Batch size</th>
<th>500 MSamples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transfer rate</td>
<td>0.1-1 GSPs</td>
</tr>
</tbody>
</table>

---

### Table 7

<table>
<thead>
<tr>
<th>Trigger rate</th>
<th>1.6 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Re-arm time</td>
<td>300 ns</td>
</tr>
</tbody>
</table>

---

**Figure 9**: Multi-record timing of use-cases. Typical parameters for illustration only. See **Table 7** for figures for the different digitizer models.
Triggered streaming mode

Figure 10 (a) is common parameters in this example. All parameters in Figure 10 have typical best case values. See Table 7 for parameters for the different models.

In Figure 10 (b) is the typical triggered streaming use-case. A continuous flow of equally spaced triggers and each record is transferred directly to the host PC. The total amount of recorded data is much larger than the ADQ DRAM. The application is optimized for transfer speed.

Figure 10 (c) is a data driven application where analog activity activates the recording, that is, level trigger. The peak load is low, so the Block RAM can be used as FIFO. The re-arm time is minimized.

<table>
<thead>
<tr>
<th>(a) Common parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
</tr>
<tr>
<td>Transfer rate</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Triggers</th>
<th>Records</th>
<th>Data transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Well scheduled triggers

<table>
<thead>
<tr>
<th>Triggers</th>
<th>Records</th>
<th>Data transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(c) Rare events and short records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak data in FIFO</td>
</tr>
<tr>
<td>Re-arm time</td>
</tr>
</tbody>
</table>

Figure 10: Triggered streaming timing of use-cases. Typical parameters for illustration only. See Table 6 for figures for the different digitizer models.

3.4 Type 2: Bandwidth reduction for RF/IF data

Bandwidth reduction is used in a band limited system with a continuous flow of data. This could be a modulated RF/IF carrier system. A typical data reduction is then to select one “RF channel” by filtering and thereby reduce the bandwidth. The data rate may then be reduced by decimation, see Figure 11.

The standard functions in the digitizers are limited to various forms of decimation and sample skip. See the datasheet of the respectively digitizer model for details on supported functions. It is recommended to use the ADQ Development Kit to implement a custom solution of bandwidth reduction inside the FPGA.
4 Multi-record mode

4.1 ADQ DRAM as data buffer

The ADQ DRAM is used for buffering up to 1 GiBytes of data at full speed (256 Mibytes on ADQ V5). ADQ DRAM access is controlled by the multi-record commands and gives full access to digitizer features such as trigger-hold off, pre-trigger, time-stamp and all trigger modes. The ADQ DRAM is the preferred data buffer when any of these conditions is fulfilled:

- The data records are large
- There is a burst of high rate triggers

For a scheduled operation like this, the total amount of data in one acquisition is limited to the size of the ADQ DRAM:

\[
\text{Required DRAM size} = \text{Number of triggers in burst} \times \text{Record length} + \text{overhead}
\]  

(2)

See Appendix C ADQ DRAM format for more information on DRAM size.

4.2 Sequence of operation

The role of the ADQ DRAM is to act as a FIFO. The low-level scheme of read and write operations limits the flexibility. The acquisition process has to be scheduled to a write phase and a read phase. A typical scheduling is:

1. Set up multi-record mode
2. Arm trigger
3. Acquire data
4. Transfer data to host PC
5. Restart from 2

The data transfer is controlled by the user application. The sequence of operations is controlled by polling flags in the digitizer. There are two different sequences available; post reading or read out while recording\(^1\). These sequences are described in the following sections.

\[\]

1. Some digitizer models support both, whereas some digitizer models only supports post reading because of memory bandwidth limitations.
Method 1: post reading

In this method, all the records are first recorded in the ADQ DRAM, then the flag is set and data may be read out, Figure 12. This method is the straight forward method and is preferred for high trigger rates.

Figure 12: Post reading of multi-record data

Method 2: Read out while recording

In this method, the number of acquired records can be read from a register. The acquired records may be transferred to the host PC before acquisition of all records is completed, Figure 13. This method is preferred if the trigger rate is low and the record size is too large for triggered streaming.

---

1. If the trigger rate is high, the advantage of Read out while recording is low. Example: trigger rate 100kHz, record sizes 10k samples, sample rate 1GSPS, read-out speed 100 MSamples/s. Then the acquisition time is 10us and the read out time is 100 us, which means only 9% reduced cycle time with the simultaneous method.
4.3 Transferring the records to the host PC

The data format in the ADQ DRAM is designed to maximize the memory speed, and to support long pre-triggers. The packing into the ADQ DRAM data format is done inside the FPGA. The unpacking is done in the PC. The ADQ DRAM set up and data handling is automatically done by the ADQAPI, and the user does not have to take care of this. However, it is important to understand the implications of these operations on the data rate. In most cases, the packing and sorting of the data takes more time than the actual transfer from the digitizer. For different digitizer models, different sets of these operations are performed by the host PC:

- Changing data format from 12b to 16b to match the data word size in the PC.
- Separating the samples for different channels into different buffers.
- Unwrapping the buffer with respect to the trigger position.

More information on the details of the memory format is available in Appendix C ADQ DRAM format.

4.4 Memory dump and off line processing.

If data is processed after the completed acquisition, this processing time can be avoided inside the real time loop. This improves the trigger rate. The data transfer to the host PC is then only a memory copy operation and the unpacking of the data is performed from the PC memory off line, Figure 14.
Figure 14: Copy data to PC memory and unpack off line.

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetTransferBuffers(1, transfer_block_size)</td>
<td>Set up a buffer for the block transfer size, larger buffer gives better performance (default is 128KiBytes)</td>
</tr>
<tr>
<td>MultiRecordSetupGP(number_of_waveforms, samples_per_record, (unsigned int *)&amp;mr_info);</td>
<td>Setup Multi-Record acquisition. Details of padding, required memory usage etc are reported in the mr_info struct.</td>
</tr>
<tr>
<td>dram_shadow = malloc(mr_info.shadow_size);</td>
<td>Allocate memory in the PC for the DRAM shadow of the ADQ DRAM.</td>
</tr>
<tr>
<td>ArmTrigger();</td>
<td>Arm the trigger and start polling GetAcquiredAll();</td>
</tr>
<tr>
<td>–</td>
<td>Trigger device with the selected trigger method</td>
</tr>
<tr>
<td>GetAcquiredAll();</td>
<td>Check that all records has been acquired.</td>
</tr>
<tr>
<td>MemoryDump(0, number_of_waveforms<em>mr_info.dram_addr_per_record , (unsigned char</em>)dram_shadow, &amp;readbytes, transfer_block_size);</td>
<td>Transfer data from the ADQ DRAM to the PC DRAM shadow area.</td>
</tr>
<tr>
<td>–</td>
<td>Store data to disk if required (data is in PC memory)</td>
</tr>
<tr>
<td>MemoryShadow(dram_shadow, max_bytes);</td>
<td>Parsing: Tell the API to use the DRAM shadow area instead of reading from ADQ. (MemoryShadow(NULL, NULL); resets the API to normal operation.)</td>
</tr>
<tr>
<td>GetData(target_buffers, samples_per_record, 2, k, 1, 0xF, 0, samples_per_record, NULL);</td>
<td>Use the standard GetData() function to parse data from the PC DRAM shadow. See Appendix C for a description of the operations. The result is available in target_buffers.</td>
</tr>
</tbody>
</table>

Table 4: API call flow chart for off line parsing
4.5 Example of ADQ412-4G multi-record

There are several partitions to take care of in the system. Below is an example of configuration, which illustrates the limitations in the parts of the system.

Assume the following

- The ADQ412-4G has 4 GSPS, 12 bits and 2 channels.
- The record length in the application is 64 KiSamples
- The trigger rate is 20 kHz (that is 32% of all samples are valid).
- The ADQ DRAM is 8 GiBits.
- The transfer rate on the data link is 3.6 GBytes/s.
- Disk write data rate is 1 GBytes/s
- Disk size is 1 TBytes.

Calculate the these parameters:

- The data rate is then 2ch * 64KiS * 20kHz * 2bytes = 5.12 GBytes/s (the 12b data is mapped to 16b). This is not theoretically possible to transmit on the link, which means that the DRAM has to be used.
- The ADQ DRAM can hold 8Gibits/12bits/2ch/(64KiSamples+overhead) = 5 k records.
- The acquisition time for the burst is 5k/20kHz = 250 ms.
- The transfer time to the PC RAM is (12/8)*5k*64k*2ch/3.6GHz = 274 ms. This is the dead time between bursts.
- The disk will be full after 1TBytes/ (1GBytes/s) = 18 minutes recording time.

Note that this is in practice a system with a burst recording, see Figure 4.

5 Triggered streaming

5.1 FPGA block RAM as buffer.

For optimized data transfer rate to the host PC, Block RAM in the FPGA is used as the data buffer. The Block RAM act as a FIFO. Writing to the FIFO is done at full speed. The average data rate is determined by the transfer rate to the host PC, that is the speed that empties the FIFO.

To avoid FIFO overflow, the data rate in to the FIFO has to be kept lower than the transfer rate to the host PC. This is done by triggered streaming. At each trigger, a record of data is written to the FIFO. If the record of data is shorter than the FIFO, it is buffered and wait for transmission to the host PC. The trigger rate is then limited by the transfer rate as:

\[
\text{Average trigger rate} = \frac{\text{Data rate to host PC}}{\text{Record length}}
\]  

(3)

There is also a limitation on the record length. The entire record must fit in the FIFO, see Table 6. Burst triggers, Figure 4, can be handled as long as there is no overflow in the FIFO. A simplified calculation of the burst triggers is:

\[
\text{Number of triggers in burst} = \frac{\text{Block RAM size}}{\text{Record length}}
\]  

(4)

To summarize, the triggered streaming mode is preferred when both these conditions are fulfilled:
The records are short
- There is a continuous flow of properly spaced triggers

The triggered streaming is supported in most trigger modes. Below is a description of external trigger and level trigger.

**External trigger**

The block diagram for triggered streaming using external trigger is in **Figure 15**. At a trigger event, one record for each channel are sent to respectively FIFOs. A header is added to the record. The mask function can select channels. The sort and multiplexer block creates a single stream of data, which is transmitted to the host PC.

This mode of operation is suitable for scheduled systems, for example, MIMO radio systems.

**Level trigger**

The block diagram for triggered streaming using level trigger is in **Figure 16**. In this example, the level trigger is set to be individual on each channel. This means that an event on channel A will produce a record on channel A only. All records from each channel are put in the respectively FIFOs. The sort and multiplexer function sends the records in time order to the host PC.

This mode of operations is suitable for individual pulse sensors in a multi-channel system.
5.2 Transfer data from ADQ Block RAM to host PC

Data is always 8 or 16 bits depending on digitizer model. When using some digitizer features like decimation or waveform averaging, the data word is 32 bits.

In the data buffer in the digitizer, there exist only unclassified data. The user has to keep track of the data format. In the digitizer any information may be put into the streaming FIFO and transferred to the PC.

Figure 16: Level trigger on individual channels.
This means that the user application has to know the start and stop of each record that is transferred and if there are any headers, etc. This method also gives the user the possibility to add any specific information into the data. See for example Figure 5, where a custom record contains various parameters from the acquired record. It is up to the user application to extract custom header, peak and area values from the data.

5.3 Example of ADQ108 using triggered streaming

There are several partitions to take care of in the system. Below is an example of configuration, which illustrates the limitations in the parts of the system.

Assume the following:

- The ADQ108 has 7 GSPS, 8 bits and 1 channel.
- The streaming FIFO (FPGA Block RAM) is 32 KiSamples. This means that the maximum data record is 32 KiSamples.
- The transfer rate on the data link is 3.6 Gbytes/s.
- Assume that the host PC can allocate 16 GiBytes of RAM for this application.

Calculate these parameters:

- The recording time for each record is 32 KiSamples / 7 GSPS = 4.7 \( \mu \)s.
- The maximum trigger rate is set by transfer rate and record size to 3.6 GHz / 32 KiSamples = 109 kHz.
- The total acquisition is limited by PC RAM to 16 GiBytes / 3.6 GHz = 5.8 s (unless data is processed or moved to disk simultaneously).

6 Data reduction for continuously streaming systems

6.1 System

Data reduction in a continuous data recording system is in practice decimation. The decimation is preceded by some signal conditioning. A typical example is a direct sampling RF receiver. The data signal processing is then typically bandpass filter, down converter, low-pass filter and decimation. The requirement on the signal bandwidth is that the required effective sample rate is lower than the transfer rate to the host.

6.2 Decimation

The data rate reduction process is a sample skip function. Sample skip by a factor of 8 means that every 8th sample is kept, and the others are discarded.

If the sample skip is preceded by a low pass filter, signal quality is preserved. This is referred to as decimation.

6.3 Buffer

The buffer has to be the Block RAM buffer in the FPGA. There is no header, only a flow of data.

6.4 Example of continuously streaming system RF system

On channel in the digitizer ADQ214 is used for direct sampling of an IF signal. The ADQ214 has an analog bandwidth of 800 MHz, and may thus be used for sub-sampling system. The system has these properties:

- The channel is 20 MHz wide.
The custom data reduction contains
- Digital mixer for down conversion
- Digital low pass (decimation filter)
- Data reduction (sample skip)
- The signal processing is done with 16 bit words, that is 2 bytes per word.

Calculate the data transfer properties:
- A minimum sample rate for ideal decimation filters is $2 \times 20 = 40$ MSPS. To relax the requirement of the decimation filter, set the sample rate to 50 MSPS. This means 100 MBytes/s.

This calculation only reflects the data transfer properties. Other aspects of designing digital radio components are not included.

7 Data link from digitizer to host PC
The ADQ V6 digitizer family supports PCIe Gen2, 8 lanes and Gen1, 4 lanes.
The ADQ V5 digitizer family supports PCIe Gen1, 4 lanes.
There is also USB 2.0 interface on all models except SDR14.
Note that the digitizers are available in several physical form factors, but the electrical interface is PCIe or USB 2.0.
The PXIe and the Micro-TCA form factors use the PCIe standard for data transfer.

7.1 Data packet sizes

Levels with packets defining sizes
All data transfer is packet based in several levels.
- Data is processed as several parallel samples. This sets a granularity on all sizes.
- The triggered data is organized in records.
- The data in general is organized in packages (ADQ14 only).
- The DRAM data word size is a sort of packet (ADQ14 only).
- The PCIe data link has TLP (transaction layer packages) for transmission
- The RAM in the host PC is set up in buffers with predefined size, that is, a sort of package.

To complete a transfer, all these buffers has to be filled and handed over (transmitted) to a receiving party, Figure 17 and Figure 18. This means that the combination of record length and data buffer size is very important.

ADQ14 data transfer system
The ADQ14 has the most advanced data transfer system. It contains a packet layer, which is only an internal design to be able to handle data flow from all channels with individual and unrelated content. Another detail for ADQ14 only is that the DRAM is used as a FIFO. Then the DRAM data word size is of importance. Everything sent to the DRAM use a number of full data words (1024 bits). This means that the ADQ14 data package has to be a multiple of 1024 bits including the header.
**ADQ V6 and ADQ V5 data transfer system**

The ADQ V6 and ADQ V5 are a subset of the ADQ14.

---

**Figure 17: Data transfer packages on ADQ V5 and ADQ V6 for streaming**

---

**Figure 18: Data transfer packages on ADQ14.**
The total record has to fit to parallel samples.

\[(RP + RH) / 64 = \text{integer}\]  \hspace{1cm} (5)
\[(RP + RH) / PP = \text{integer} = \text{NofP}\]  \hspace{1cm} (6)
\[PP / 64 = \text{integer}\]  \hspace{1cm} (7)
\[(PP + PH) / DW = \text{integer}\]  \hspace{1cm} (8)
\[(PP + PH) / TLP = \text{integer}\]  \hspace{1cm} (9)
\[B / ( (PP + PH) \times \text{NofP} \times \text{integer} ) = \text{integer}\]  \hspace{1cm} (10)

To be completed ...

---

**Table 5: Packet size parameters**

<table>
<thead>
<tr>
<th>LEVEL</th>
<th>ADQ V5</th>
<th></th>
<th></th>
<th>ADQ V6</th>
<th></th>
<th></th>
<th>ADQ15</th>
<th></th>
<th></th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MIN</td>
<td>MAX</td>
<td>STEP</td>
<td>MIN</td>
<td>MAX</td>
<td>STEP</td>
<td>MIN</td>
<td>MAX</td>
<td>STEP</td>
<td></td>
</tr>
<tr>
<td>Record pay-load (RP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record header (RH)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package pay-load (PP)</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package header (PH)</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DRAM word (DW)</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>--</td>
<td>1024</td>
<td>1024</td>
<td>1024</td>
<td>Set in PC</td>
</tr>
<tr>
<td>TLP (TLP)</td>
<td>128</td>
<td>256</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>128</td>
<td>128</td>
<td>256</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>Buffer (B)</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.2 Command sequence for data transfer

There are different command sequences depending on the method selected for implementing the user software.

7.2.1 Multi-record

When using Multi-record mode, the data is acquired into the RAM of the digitizer and is resident there waiting for being transferred to the RAM of the host PC, Figure 19.

The user polls to see if a record is available and then sets up a data transfer.

When the data transfer is completed the pointer to the buffer with the data is handed over to the user.

![Data transfer commands for Multi-Record.](image-url)

Figure 19: Data transfer commands for Multi-Record.
7.2.2 Triggered Streaming

When using triggered streaming mode, the data is acquired into the RAM of the digitizer and then transferred to the PC RAM in an automatic sequence. The data is resident in the PC RAM, Figure 20.

The user polls to see if a record is available in the PC RAM. When the data is available the pointer to the buffer with the data is handed over to the user.

The sequence of commands is basically the same as for the multi-record transfer in Section 7.2.1. But the data waits in the PC RAM instead of the RAM of the digitizer. This sequencing has a potential of being much faster than the multi-record approach.

Figure 20: Data transfer commands for triggered streaming.
7.2.3 Triggered Streaming Multi-Treaded

Important notice: The ADQAPI is not completely thread safe. Below is a method for multi-threading that works if it is handled with care.

The system is divided into two threads. One thread controls the digitizer and one thread post-process the data. Note that only the first thread talks to the ADQ hardware. Thus the interface to the hardware is a single thread.

The data is acquired into the RAM and then transferred from the digitizer to the PC RAM in an automatic sequence. The data is resident in the PC_RAM, Figure 21. This is the same as for triggered streaming, Section 7.2.2.

The second thread is started and asks for a completed acquisition. By setting the time out very large, the command is on hold waiting until the acquisition is completed. As soon as data is available, post-processing is started. The post processing thread notifies the first thread that a new acquisition can be prepared.

![Diagram](image-url)

Figure 21: Data transfer commands for Multi-Threaded triggered streaming.
8 Setting up the host PC for streaming

8.1 Introduction

Data from the digitizer is stored in the host PC RAM. The PC RAM is divided into several buffers where data may be written. The user gets a pointer to a filled buffer.

From the host PC RAM, data is either consumed by the application or stored to disk. See Section 10 for more details.

8.2 Principle of operation

At start-up, the ADQAPI reserves data buffers for the digitizer. The buffers are used for sending data from the digitizer to the user application. The sequence of data transfer is illustrated in Figure 22. (See Section 9 for description of ADQAPI_preallocate.exe)

8.3 Setting buffer size

When the ADQ control unit is created, a default set of 8 instances of 128 KiByte large buffers are allocated. The user may allocate larger buffers if necessary. Too small buffers cause a high interrupt rate, which reduces overall systems performance. This is of course very systems dependent. An indication is to keep the interrupt rate below a few hundred hertz. There is in principle no upper limit to the size of a buffer. However, the ADQAPI requires physically continuous memory, which may limit the amount of usable memory even while free memory is available in the virtual page pool.

8.4 Re-allocate RAM

Each time the ADQ control unit is deleted, the ADQAPI releases the buffers. This happens, for example, when the digitizer is re-started or when the ADQ application is stopped. When a new ADQ control unit is created, new buffers are allocated. However, if the PC operates for a long time, the memory gets fragmented and thus large buffer may be hard to allocate. This is solved using a memory pre-allocation application, see Section 9.

8.5 Example on allocating RAM

Assume the following situation using ADQ108

- Sample rate 7 GSPS
- 8 bits resolution, that is 1 Byte/sample
- Record size 128 KiSamples
- Burst trigger rate 1 kHz
- One burst is 512 triggers
- The measurement is repeated 2048 times

Calculate parameters for the application

- The FPGA block RAM buffer is large enough for 128 KiSamples; use triggered streaming.
- Data rate on the link to the host is 128 KiSamples * 1 kHz = 131 MBytes/s. This is feasible over PCIe.
- Interrupt rate at typically <100 Hz requires a buffer size of >1.25 MiBytes. 
  \[ (131 \text{ MBytes/s} \times (10^6 /1024^2) /100 \text{ Hz} = 1.25 \text{ MiBytes}) \]
- One burst is 128 KiSamples * 512 trigger = 64 MiBytes of data
- Total measurement is 128 KiSamples * 512 trigger * 2048 repetitions = 128 GiBytes.
Figure 22: Flow for streaming application.

**Systems requirement**

- The total burst of 64 MiBytes of data has to fit into the host PC RAM.
• The total measurement of 128 GiBytes of data has to be stored on disk.
• The time between the burst is limited by the time it takes to transfer 64 MiBytes to the disk.

9 Pre-allocation application

9.1 Introduction

When the PC operates for a long time, the memory gets fragmented. For efficient data transfer, there is a memory pre-allocation application that reserves RAM space for the digitizer application. When a digitizer application tries to allocate DMA buffers in the RAM, it will first check if there are any pre-allocated buffers, and use these if possible. See Appendix A Reference guide to ADQAPI_preallocate.exe for detailed information.

9.2 Starting the pre-allocation application

It is recommended to add this application with console arguments to the Windows Task Scheduler, and set to "run before logon".

9.3 Setting RAM size

For a scheduled real-time operation, the amount of required RAM is well defined. Then set the total amount of RAM to this value. The RAM amount will be separated in a set of buffers. See Section 8.3 on how to set the buffer size. With this method, there will always be RAM buffers available for the data from the digitizer.

If the behavior is not well defined, it may be difficult to foresee how much pre-allocated memory that is required. If the pre-allocated RAM is too small for a momentary peak load, the ADQAPI will automatically allocate new RAM. This automatic allocation will also maintain backwards-compatibility.

9.4 Status check

Start the pre-allocation application at any time and use the console menu to check status or free the allocated buffers.

10 Consuming the data in the host PC

10.1 Extracting parameters

Eventually a limited set of parameters are typically extracted from the raw data, that is, a peak height, an energy measure, or any other parameter. This process will free the buffers to be filled with new data. There are typically two ways to handle this:

Limited acquisition time

The amount of data from the entire acquisition is limited to an amount that fits into the host PC RAM. Then there are no fundamental requirements on the processing of the data. The processing time sets the minimum time before the sequence can start over again.

Unlimited acquisition time

The user has to consume the data in the same rate as it is written into the host PC RAM. This may be an indirect limiting factor on, for example, trigger frequency in the system.

10.2 Storing raw data to disk

Data coming over the data link is always stored in the host PC RAM first. From that, it is copied to disk by the user application. The ADQAPI delivers a data pointer to the user, where data...
is available. Copying the data to disk will free the buffers. In general, the digitizers puts high requirements on the write speed to the disk.

**Appendix A Reference guide to ADQAPI_preallocate.exe**

**Installation:**

- Install the ADQAPI and drivers using ADQ-setup.exe. This will install the file ADQAPI_preallocate.exe.
- The application is found in the installer C:\Program Files\SP Devices\Preallocate or C:\Program Files\SP Devices\Preallocate_x64. Place the ADQAPI_preallocate.exe in preferred folder.

**Running:**

- Run ADQAPI_preallocate.exe
- It should display a menu allowing you to allocate or free buffers.
  - If it does not open correctly, run it from a cmd.exe console, and check for any error messages.
  - This application can also check the amount of allocated and used buffers, however currently it does not run simultaneously to any application using the ADQ boards.
  - Instead of using the menu, it can be started by using arguments. If so ADQAPI_preallocate will terminate when finished, it will not stay resident. See below for details on how to use this.

**Running automatically at boot:**

- Open Control Panel → System and Security → Administrative Tools → Schedule Tasks
- Action → Create Task
  - Name: ADQAPI_preallocate
  - Run whether user is logged on or not
  - Triggers → New…
    - At Startup
  - Actions → New…
    - Start a program
      - C:\<your folder>\ADQAPI_preallocate.exe
      - Add arguments <number of buffers>:<buffer size in bytes>.
        - For example 64:8388608 means 64 buffers of 8 Mbytes each.
        - 8 Mbytes buffer size typically gives good performance.
        - Buffer size can be set in steps of 4 Kbytes.
  - Conditions
    - Set as you please
  - Settings
    - Set as you please
- Restart the computer
• When restarted, run ADQAPI_preallocate and check the preallocation status.
  • It may not be able to allocate all of the requested buffers, it will then allocate as many as were possible at startup.

Running the digitizer application:
• Any application using the new ADQAPI.dll will automatically use the reserved buffers, when available.
• If any digitizer application tries to use more than the pre-allocated buffers, it will try to allocate additional buffers. This also means that it will transparently fall back to the old behavior, should no pre-allocated buffers be available.
## Appendix B Parameters for the different digitizer models

<table>
<thead>
<tr>
<th>MODEL</th>
<th>STREAMING FIFO SIZE [KISAMPLES]</th>
<th>PRE-TRIGGER [SAMPLES]</th>
<th>DATA LINK SUSTAINED(^1) [MBYTES/S]</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADQ108</td>
<td>128</td>
<td>16384</td>
<td>3600</td>
<td></td>
</tr>
<tr>
<td>ADQ208</td>
<td>64</td>
<td>16384</td>
<td>1800</td>
<td>Per channel</td>
</tr>
<tr>
<td>ADQ412</td>
<td>128 / 64</td>
<td>8192 / 4096</td>
<td>1800 / 900</td>
<td>Per channel: 2 channels / 4 channels</td>
</tr>
<tr>
<td>ADQ412-3G</td>
<td>128 / 64</td>
<td>8192 / 4096</td>
<td>1800 / 900</td>
<td>Per channel: 2 channels / 4 channels</td>
</tr>
<tr>
<td>ADQ412-4G</td>
<td>128 / 64</td>
<td>8192 / 4096</td>
<td>1800 / 900</td>
<td>Per channel: 2 channels / 4 channels</td>
</tr>
<tr>
<td>SDR14</td>
<td>32</td>
<td>4096</td>
<td>1800</td>
<td>Per channel</td>
</tr>
<tr>
<td>ADQ1600</td>
<td>192</td>
<td>4096</td>
<td>3600</td>
<td></td>
</tr>
</tbody>
</table>

### ADQ V6 digitizer family

<table>
<thead>
<tr>
<th>MODEL</th>
<th>DRAM [BYTES]</th>
<th>DRAM(^1) [MISAMPLES/CHANNEL]</th>
<th>RE-ARM TIME [NS]</th>
<th>DATA LINK FROM DRAM(^2) [MBYTES/S]</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADQ108</td>
<td>1 Gi</td>
<td>1024</td>
<td>285</td>
<td>550</td>
<td></td>
</tr>
<tr>
<td>ADQ208</td>
<td>1 Gi</td>
<td>512</td>
<td>250</td>
<td>400 / 550</td>
<td>2 channels / 1 channel</td>
</tr>
<tr>
<td>ADQ412</td>
<td>1 Gi</td>
<td>340/170</td>
<td>500</td>
<td>55 / 55</td>
<td>2 channels / 4 channels</td>
</tr>
<tr>
<td>ADQ412-3G</td>
<td>1 Gi</td>
<td>340/170</td>
<td>275</td>
<td>55 / 55</td>
<td>2 channels / 4 channels</td>
</tr>
<tr>
<td>ADQ412-4G</td>
<td>1 Gi</td>
<td>340/170</td>
<td>250</td>
<td>55 / 55</td>
<td>2 channels / 4 channels</td>
</tr>
<tr>
<td>SDR14</td>
<td>1 Gi</td>
<td>128</td>
<td>310</td>
<td>120</td>
<td></td>
</tr>
<tr>
<td>ADQ1600</td>
<td>1 Gi</td>
<td>512</td>
<td>310</td>
<td>275</td>
<td></td>
</tr>
</tbody>
</table>

### ADQ V5 digitizer family

<table>
<thead>
<tr>
<th>MODEL</th>
<th>DRAM [BYTES]</th>
<th>DRAM(^1) [MISAMPLES/CHANNEL]</th>
<th>RE-ARM TIME [NS]</th>
<th>DATA LINK FROM DRAM(^2) [MBYTES/S]</th>
<th>COMMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADQ112</td>
<td>256 Mi</td>
<td>170</td>
<td>440</td>
<td>180</td>
<td>14 bits data mapped to 16 bits words</td>
</tr>
<tr>
<td>ADQ114</td>
<td>256 Mi</td>
<td>128</td>
<td>600</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>ADQ212</td>
<td>256 Mi</td>
<td>85</td>
<td>440</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>ADQ214</td>
<td>256 Mi</td>
<td>64</td>
<td>600</td>
<td>180</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6: Data transfer parameters triggered streaming\(^3\)

1. Sustained means average transfer time without loss of data until the buffer in the host PC RAM is full. This is achieved in a reference PC set-up.
2. ADQ V6 values are for PCIe Gen2 x8 interface.
3. Parameters are to be confirmed

### Table 7: Data transfer parameters multi-record\(^4\)

1. Record headers requires some space. The effective number of samples is thus lower.
2. Sustained means average transfer capacity without loss of data until the buffer in the host PC RAM is full. This is achieved in a reference PC set-up.
3. Best case using the GetData() function. Limited by unpacking in the PC, see Appendix A.
4. Parameters are to be confirmed
Appendix C ADQ DRAM format

C.1 Data records in the ADQ DRAM

In the multi-record mode, the data is stored in records in the ADQ DRAM. Each record contains a 64 bytes header. There is also up to 64 bytes data overhead to handle mapping to DRAM data bus. The overhead parameter is thus 64 – 128 bytes/record. Figure 23 illustrates the relation between record size and number of records for an ADQ412.

![Figure 23: Number of records in DRAM on ADQ412](image)

C.2 Packing data in the ADQ DRAM

The memory is organized in large ADQ DRAM words which can hold several data words. The ADQ DRAM words are grouped into pages. One page size is the minimum size for one data record. The positions in memory that are not used for data will be overhead.

Some of the digitizers deliver 12 bit data, which would waste 4 bits if stored in a 16 bit word. To reduce bandwidth requirement to the ADQ DRAM, data is packed into the larger ADQ DRAM words without wasted bits. This also allows for larger records.

To enable a large pre-trigger, the ADQ DRAM operates as a large circular buffer. The trigger position might thus be anywhere in the buffer.

The data from the different channels has to be written simultaneously to the ADQ DRAM. The result is that the channels are interleaved in the ADQ DRAM. Figure 24 illustrates the memory organization.

---

1. See 12-0768 for information about the record header. The record header is handled by the ADQ API and the user does normally not have to care about the header.
2. The page size is defined by a combination of parameters and is different on all models.
**C.3 Unpacking data from the ADQ DRAM**

When reading the data from the ADQ DRAM, it is first mapped to 8, 16 or 32 bits words, which can be used in the PC. This is done automatically in the ADQ API commands for getting data records from the digitizer. Data is also separated into the different channels. This un-packing requires a lot of resources from the CPU and may put a lower limit to the data transfer than the data link.

Finding the position of the record within the data is the second operation, Trigger information is read from the header by the ADQAPI. The record is extracted with respect to the trigger position before it is handed over to the user. The unused data is also cut and the resulting data has the correct length.

**Figure 24** illustrates these steps of operations.

**Appendix D USB freezing**

The USB version of the ADQ digitizers may sometimes be disconnected. This is caused by unreliable operation in a USB hub/port when multiple USB devices (not necessary ADQ digitizers) are connected to one PC.
This can, for example, be detected by that CollectDataNextPage() returns 0 until the whole application is shut down and restarted. Checking device manager on Windows will however show that the ADQ is still connected because the driver can automatically re-detect it. But the running application has lost all connection with the device.

This is not uncommon to happen on Windows when the power plan for Windows is set to enable the option "USB selective suspend". This option can be turned off according to this link:

http://www.techsupportalert.com/content/how-fix-annoying-windows-usb-problem.htm

However, on some systems this option can turn itself back on again. The it is required to make a change in the windows registry according to here:


and here:

http://www.justanswer.com/computer/2s6pl-permanently-disable-usb-selective-suspend-re-enables.html

Also set the PC on "High Performance" power plan for long term measurements. Streaming performance is highly depending on the performance of the host PC.

Appendix E    Acquisition modes example code

The following examples are available for each acquisition mode.

<table>
<thead>
<tr>
<th>MODE</th>
<th>EXAMPLE</th>
<th>SOFTWARE COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Record</td>
<td>ADQAPI_simple_example.zip</td>
<td>MultiRecordSetup</td>
</tr>
<tr>
<td></td>
<td>ADQAPI_example.zip</td>
<td>MultiRecordClose</td>
</tr>
<tr>
<td></td>
<td>ADQAPI_example_Cplusplus.zip</td>
<td>GetAcquired</td>
</tr>
<tr>
<td></td>
<td>ADQXXX_multitrig_example_script.m</td>
<td>GetAcquiredAll</td>
</tr>
<tr>
<td></td>
<td>ADQ_multirecord_example.py</td>
<td>GetData</td>
</tr>
<tr>
<td></td>
<td>ADQ214_example.py</td>
<td>(CollectRecord) – not recommended</td>
</tr>
<tr>
<td></td>
<td>SDR14_example.py</td>
<td></td>
</tr>
<tr>
<td>Continuous Streaming</td>
<td>ADQAPI_example.zip</td>
<td>SetTransferBuffers</td>
</tr>
<tr>
<td></td>
<td>ADQAPI_simple_streaming_example.zip</td>
<td>SetDataFormat</td>
</tr>
<tr>
<td></td>
<td>ADQ412_streaming_example_script.m</td>
<td>SetStreamStatus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StartStreaming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>StopStreaming</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetTransferBufferStatus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CollectDataNextPage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GetStreamOverflow</td>
</tr>
</tbody>
</table>

Triggered streaming (only ADQ412)

Table 8:   Examples available
<table>
<thead>
<tr>
<th>MODE</th>
<th>EXAMPLE</th>
<th>SOFTWARE COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADQ412_ts_example_script.m</td>
<td>SetTransferBuffers</td>
</tr>
<tr>
<td></td>
<td>ADQAPI_example.zip</td>
<td>SetTransferBuffers</td>
</tr>
</tbody>
</table>

Continuous MultiRecord (only ADQ V6 family)

<table>
<thead>
<tr>
<th>MODE</th>
<th>EXAMPLE</th>
<th>SOFTWARE COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADQAPI_example_RecordStreaming.zip</td>
<td>MultiRecordSetupGP</td>
</tr>
<tr>
<td></td>
<td>ADQ412_cmr_example_script.m</td>
<td>MultiRecordClose</td>
</tr>
</tbody>
</table>

Waveform Averaging

<table>
<thead>
<tr>
<th>MODE</th>
<th>EXAMPLE</th>
<th>SOFTWARE COMMANDS</th>
</tr>
</thead>
</table>

Off-line parsing

<table>
<thead>
<tr>
<th>MODE</th>
<th>EXAMPLE</th>
<th>SOFTWARE COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADQAPI_OfflineParsing.zip</td>
<td>SetTransferBuffers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MultiRecordClose</td>
</tr>
</tbody>
</table>

Table 8: Examples available
Important Information

Signal Processing Devices Sweden AB (SP Devices) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to SP Devices’ general terms and conditions supplied at the time of order acknowledgment.

SP Devices warrants that each product will be free of defects in materials and workmanship, and conform to specifications set forth in published data sheets, for a period of one (1) year. The warranty commences on the date the product is shipped by SP Devices. SP Devices’ sole liability and responsibility under this warranty is to repair or replace any product which is returned to it by Buyer and which SP Devices determines does not conform to the warranty. Product returned to SP Devices for warranty service will be shipped to SP Devices at Buyer’s expense and will be returned to Buyer at SP Devices’ expense. SP Devices will have no obligation under this warranty for any products which (i) has been improperly installed; (ii) has been used other than as recommended in SP Devices’ installation or operation instructions or specifications; or (iii) has been repaired, altered or modified by entities other than SP Devices. The warranty of replacement products shall terminate with the warranty of the product. Buyer shall not return any products for any reason without the prior written authorization of SP Devices.

In no event shall SP Devices be liable for any damages arising out of or related to this document or the information contained in it.

SP DEVICES’ EXPRESS WARRANTY TO BUYER CONSTITUTES SP DEVICES’ SOLE LIABILITY AND THE BUYER’S SOLE REMEDY WITH RESPECT TO THE PRODUCTS AND IS IN LIEU OF ALL OTHER WARRANTIES, LIABILITIES AND REMEDIES. EXCEPT AS THUS PROVIDED, SP DEVICES DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT.

SP DEVICES DOES NOT INDEMNIFY, NOR HOLD THE BUYER HARMLESS, AGAINST ANY LIABILITIES, LOSSES, DAMAGES AND EXPENSES (INCLUDING ATTORNEY’S FEES) RELATING TO ANY CLAIMS WHATSOEVER. IN NO EVENT SHALL SP DEVICES BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFIT, LOST DATA AND THE LIKE, DUE TO ANY CAUSE WHATSOEVER. NO SUIT OR ACTION SHALL BE BROUGHT AGAINST SP DEVICES MORE THAN ONE YEAR AFTER THE RELATED CAUSE OF ACTION HAS ACCRUED. IN NO EVENT SHALL THE ACCRUED TOTAL LIABILITY OF SP DEVICES FROM ANY LAWSUIT, CLAIM, WARRANTY OR INDEMNITY EXCEED THE AGGREGATE SUM PAID TO SP BY BUYER UNDER THE ORDER THAT GIVES RISE TO SUCH LAWSUIT, CLAIM, WARRANTY OR INDEMNITY.

Worldwide Sales and Technical Support

www.spdevices.com

SP Devices Corporate Headquarters

Teknikringen 6
SE-583 30 Linköping
Sweden

Phone: +46 (0)13 465 0600
Fax: +46 (0)13 991 3044
Email: info@spdevices.com

Copyright © 2015 Signal Processing Devices Sweden AB.
All rights reserved, including those to reproduce this publication or parts thereof in any form without permission in writing from SP Devices.